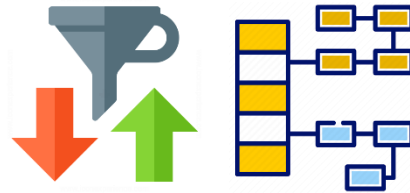


حل تمرین و نمونه سوال Hash و Sort

ساختمان داده و الگوریتم - پائیز ۱۴۰۲



دستگرمی

با فرض داشتن یک آرایه، چگونه می‌توان جایگشتی از این آرایه پیدا کرد که در آن مجموع همه $|A[i] - A[i-1]|$ ها کمینه شود؟



دستگرمی

با فرض داشتن یک آرایه، چگونه می‌توان جایگشتی از این آرایه پیدا کرد که در آن مجموع همه $|A[i] - A[i-1]|$ ها کمینه شود؟

پاسخ:

با مرتب کردن آرایه به جایگشت مورد نظر می‌رسیم که زمان اجرای $O(n \log n)$ دارد.



سوال اول

الگوریتمی با زمان اجرای $O(n \log n)$ طراحی کنید که با گرفتن آرایه‌ای به طول n بعنوان ورودی، تعداد نابجایی‌های داخل آرایه را تعیین کند. (نابجایی برای دو عنصر یعنی به ازای $i < j$ ، $A[i] > A[j]$ باشد).



سوال اول

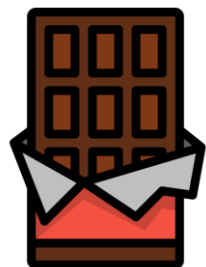
الگوریتمی با زمان اجرای $O(n \log n)$ طراحی کنید که با گرفتن آرایه‌ای به طول n بعنوان ورودی، تعداد نابجایی‌های داخل آرایه را تعیین کند. (نابجایی برای دو عنصر یعنی به ازای $i < j$ ، $A[i] > A[j]$ باشد.)

تعداد نابجایی‌های درون کل آرایه برابر است با تعداد نابجایی‌های نیمه راست آرایه + تعداد نابجایی‌های نیمه چپ آرایه + تعداد نابجایی‌هایی که i در نیمه چپ و j در نیمه راست است. برای محاسبه این مقدار از ایده merge sort استفاده می‌کنیم با این تفاوت که هنگام merge کردن دو زیر آرایه چپ و راست، هروقت عنصری از زیر آرایه راست را در آرایه اصلی گذاشتیم، به تعداد عناصر باقی‌مانده در زیر آرایه چپ به تعداد نابجایی‌ها اضافه می‌شود. زمان اجرا مانند الگوریتم merge sort و برابر $O(n \log n)$ است.



سوال دوم

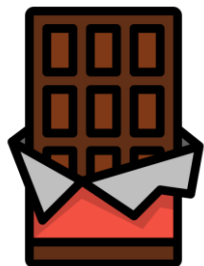
محمد تعداد k فرزند دارد و می‌خواهد به فروشگاه رفته و برای آن‌ها شکلات بخرد. در فروشگاه n جعبه وجود دارد که هر جعبه تعدادی شکلات دارد. از آنجا که باید عدالت بین فرزندان را رعایت کرد او می‌خواهد بداند می‌تواند چند جعبه را به نحوی انتخاب کند که مجموع شکلات‌های آن‌ها بر k بخش پذیر باشد. الگوریتمی با پیچیدگی $O(kn)$ ارائه دهید که وجود این ویژگی را در فروشگاه بررسی کند.



سوال دوم

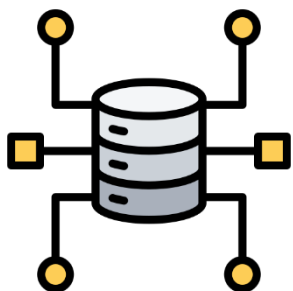
محمد تعداد k فرزند دارد و می‌خواهد به فروشگاه رفته و برای آن ها شکلات بخرد. در فروشگاه n جعبه وجود دارد که هر جعبه تعدادی شکلات دارد. از آنجا که باید عدالت بین فرزندان را رعایت کرد او می‌خواهد بداند می‌تواند چند جعبه را به نحوی انتخاب کند که مجموع شکلات های آن ها بر k بخش پذیر باشد. الگوریتمی با پیچیدگی $O(kn)$ ارائه دهید که وجود این ویژگی را در فروشگاه بررسی کند.

در ابتدا یک آرایه به طول k با مقادیر اولیه صفر را مقداردهی می‌کنیم، در نظر می‌گیریم جایگاه i ام این آرایه نشان دهنده آن است که آیا تا جایی که پیمایش کرده ایم زیرمجموعه ای از عناصر وجود دارد که جمع آن ها به پیمانه k برابر ۱ شود یا خیر. روی آرایه جعبه‌ها پیمایش می‌کنیم به هر عنصر جدید مانند c که می‌رسیم، روی آرایه k تایی پیمایش می‌کنیم و هر خانه ای مانند i که true باشد به این معنا است که باید خانه $(c + i) \% k$ در آرایه هم ۱ شود البته آرایه k تایی را بعد از اینکه یکبار روی آن پیمایش کردیم بروزرسانی می‌کنیم. الگوریتم تا جایی که خانه ۰ ام آرایه برابر ۱ شود ادامه می‌دهیم.

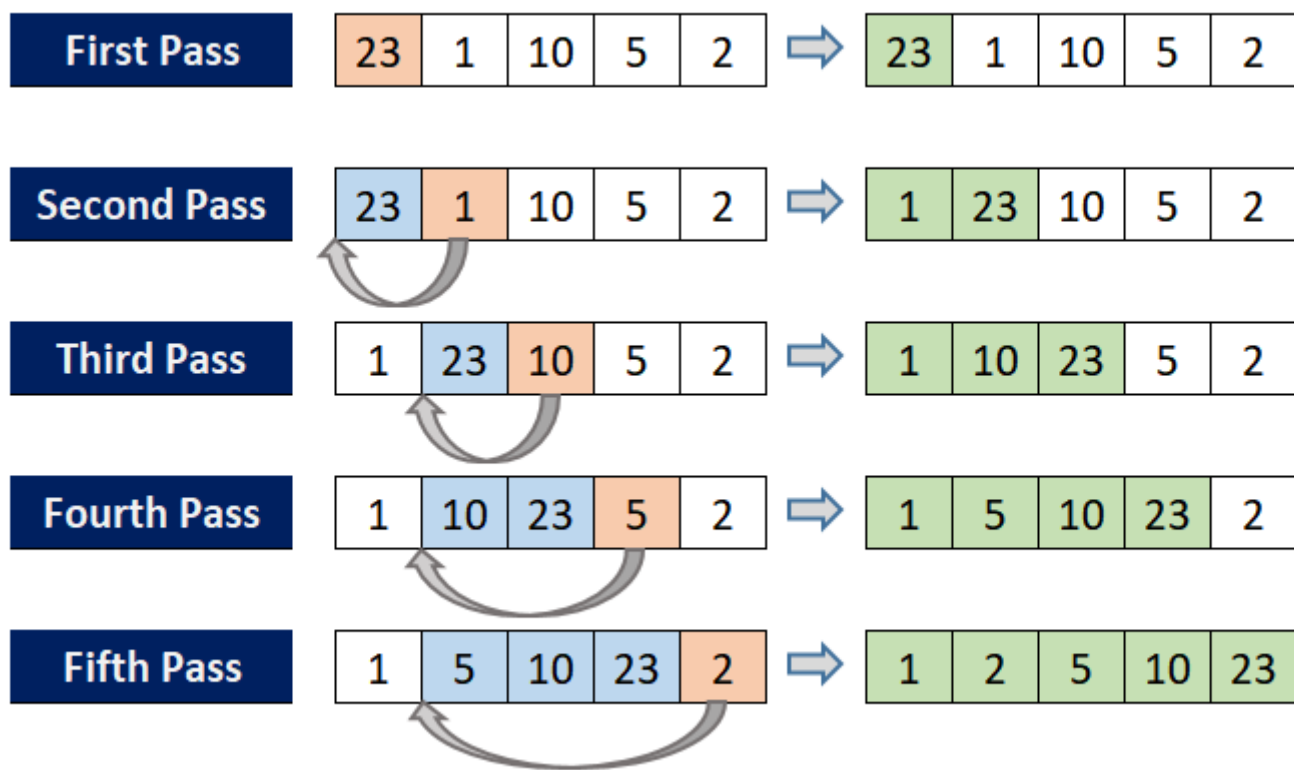


سوال سوم

برای هر یک از شرایط زیر، یک الگوریتم sort انتخاب کنید. (selection sort, insertion sort, merge sort)







خلاصه‌ای از الگوریتم های مرتب سازی مرتب سازی درجی (Insertion sort)



Insertion sort	شاخص
بله	Comprative (مقایسه ای)
بله	In-place (درجا)
بله	Internal (داخلی)
می تواند پایدار باشد.	Stable (پایدار)
$O(n^2)$	Worst-case order

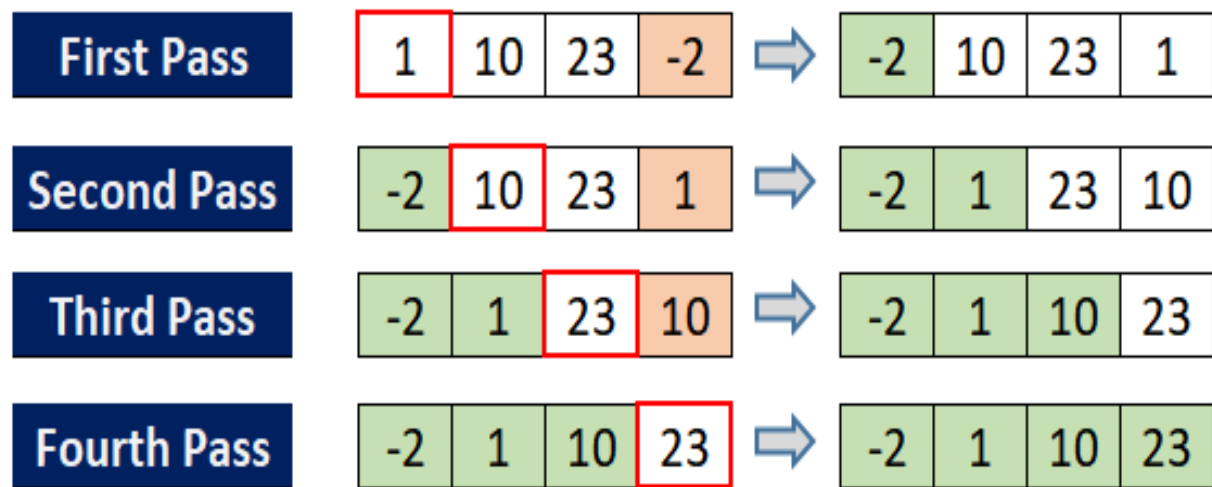
خلاصه‌ای از الگوریتم های مرتب سازی مرتب سازی حبابی (Bubble sort)

Bubble sort example

Initial	<table style="display: inline-table; border-collapse: collapse;"><tr><td style="padding: 5px 15px;">5</td><td style="padding: 5px 15px;">3</td><td style="padding: 5px 15px;">8</td><td style="padding: 5px 15px;">4</td><td style="padding: 5px 15px;">6</td></tr></table>	5	3	8	4	6	Initial Unsorted array
5	3	8	4	6			
Step 1	<table style="display: inline-table; border-collapse: collapse;"><tr><td style="padding: 5px 15px;">5</td><td style="padding: 5px 15px;">3</td><td style="padding: 5px 15px;">8</td><td style="padding: 5px 15px;">4</td><td style="padding: 5px 15px;">6</td></tr></table> 	5	3	8	4	6	Compare 1 st and 2 nd (Swap)
5	3	8	4	6			
Step 2	<table style="display: inline-table; border-collapse: collapse;"><tr><td style="padding: 5px 15px;">3</td><td style="padding: 5px 15px;">5</td><td style="padding: 5px 15px;">8</td><td style="padding: 5px 15px;">4</td><td style="padding: 5px 15px;">6</td></tr></table> 	3	5	8	4	6	Compare 2 nd and 3 rd (Do not Swap)
3	5	8	4	6			
Step 3	<table style="display: inline-table; border-collapse: collapse;"><tr><td style="padding: 5px 15px;">3</td><td style="padding: 5px 15px;">5</td><td style="padding: 5px 15px;">8</td><td style="padding: 5px 15px;">4</td><td style="padding: 5px 15px;">6</td></tr></table> 	3	5	8	4	6	Compare 3 rd and 4 th (Swap)
3	5	8	4	6			
Step 4	<table style="display: inline-table; border-collapse: collapse;"><tr><td style="padding: 5px 15px;">3</td><td style="padding: 5px 15px;">5</td><td style="padding: 5px 15px;">4</td><td style="padding: 5px 15px;">8</td><td style="padding: 5px 15px;">6</td></tr></table> 	3	5	4	8	6	Compare 4 th and 5 th (Swap)
3	5	4	8	6			
Step 5	<table style="display: inline-table; border-collapse: collapse;"><tr><td style="padding: 5px 15px;">3</td><td style="padding: 5px 15px;">5</td><td style="padding: 5px 15px;">4</td><td style="padding: 5px 15px;">6</td><td style="padding: 5px 15px;">8</td></tr></table>	3	5	4	6	8	Repeat Step 1-5 until no more swaps required
3	5	4	6	8			

Insertion sort & Bubble sort	شاخص
مقایسه‌ای	Comprative (مقایسه ای)
درجا	In-place (درجا)
داخلی	Internal (داخلی)
می تواند پایدار باشد.	Stable (پایدار)
$O(n^2)$	Worst-case order

خلاصه‌ای از الگوریتم‌های مرتب‌سازی مرتب‌سازی انتخابی (Selection sort)



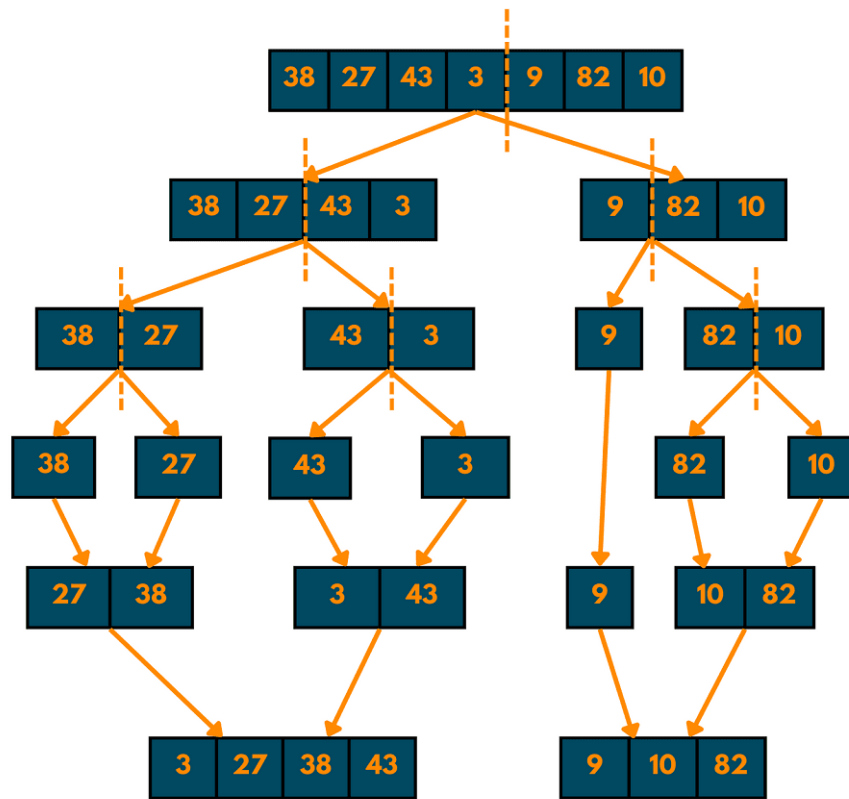
Sorted Array

Smallest number in unsorted array

Correct position for smallest number in unsorted array

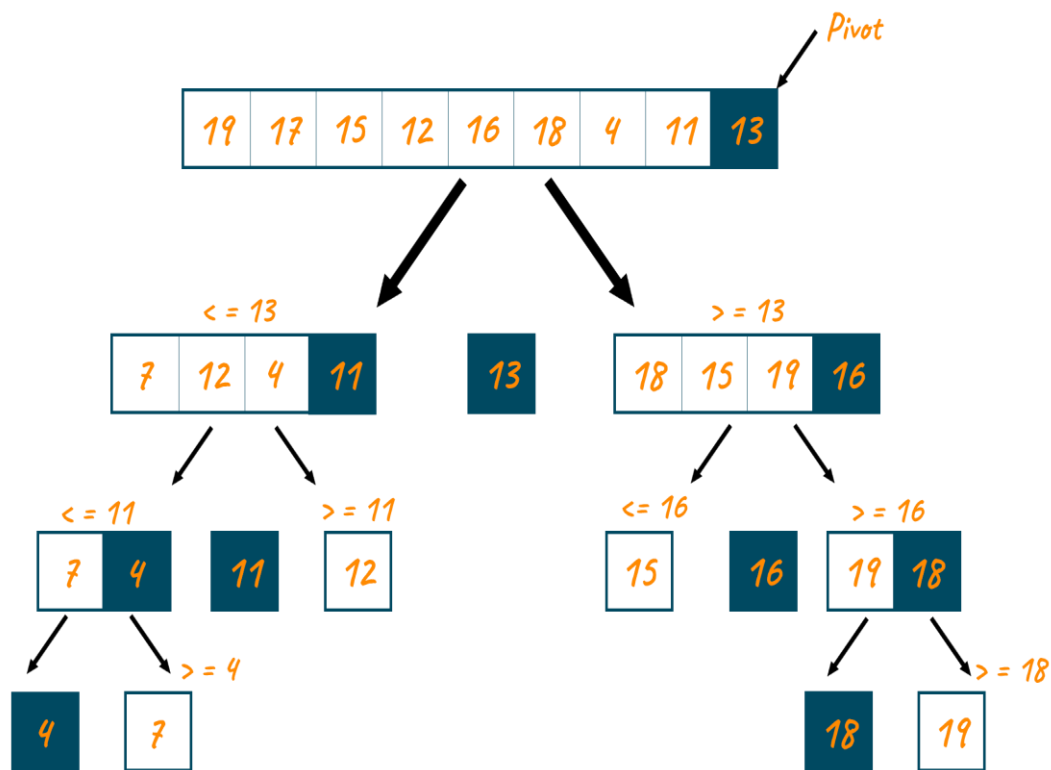
Insertion sort & Bubble sort & Selection sort	شاخص
مقایسه‌ای	Comprative (مقایسه‌ای)
درجا	In-place (درجا)
داخلی	Internal (داخلی)
می تواند پایدار باشد.	Stable (پایدار)
$O(n^2)$	Worst-case order

خلاصه‌ای از الگوریتم‌های مرتب‌سازی مرتب‌سازی ادغامی (Merge sort)



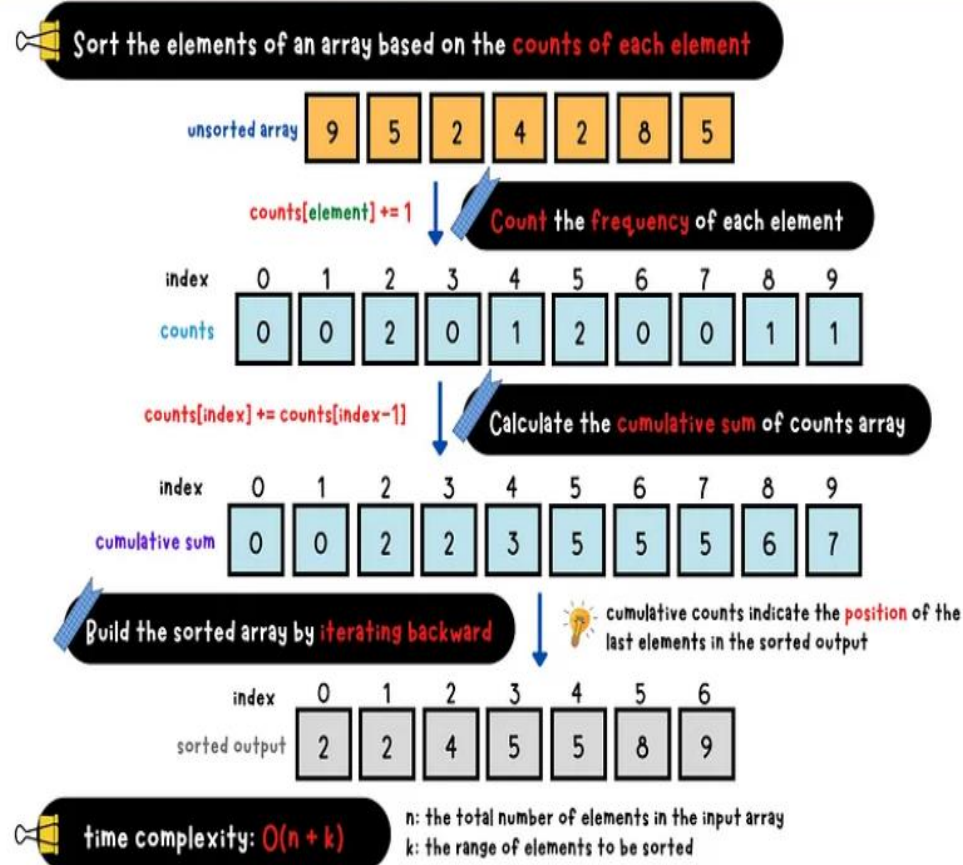
Merge sort	Insertion sort & Bubble sort & Selection sort	شاخص
مقایسه‌ای	مقایسه‌ای	Comparative (مقایسه‌ای)
برون‌جا	درجا	In-place (درجا)
داخلی	داخلی	Internal (داخلی)
پایدار	می‌تواند پایدار باشد.	Stable (پایدار)
$O(n \log n)$	$O(n^2)$	Worst-case order

خلاصه‌ای از الگوریتم‌های مرتب‌سازی مرتب‌سازی سریع (Quick sort)



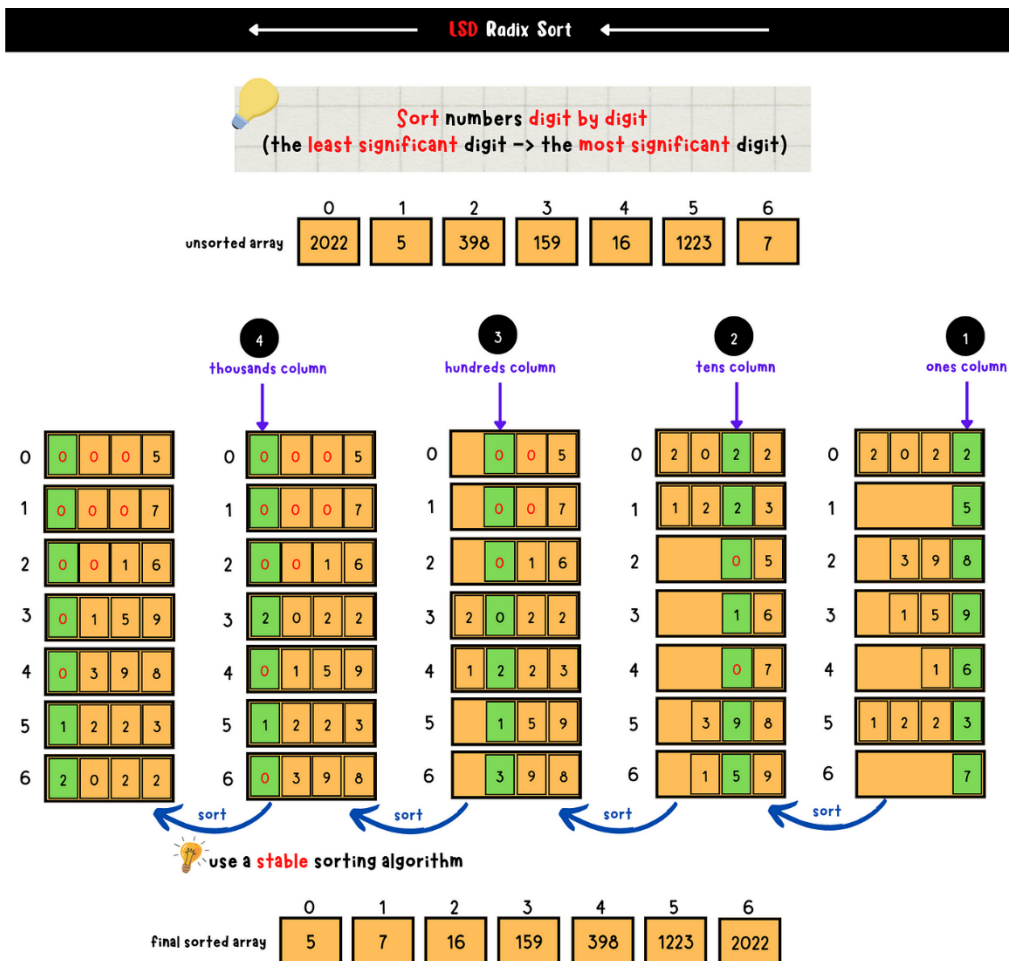
Quick sort	Merge sort	Insertion sort & Bubble sort & Selection sort	شاخص
مقایسه‌ای	مقایسه‌ای	مقایسه‌ای	Comparative (مقایسه‌ای)
درجا/برون‌جا	برون‌جا	درجا	In-place (درجا)
داخلی	داخلی	داخلی	Internal (داخلی)
ناپایدار	پایدار	می‌تواند پایدار باشد.	Stable (پایدار)
$O(n^2)$	$O(n \log n)$	$O(n^2)$	Worst-case order
$O(n \log n)$			
$O(n \log n)$			

خلاصه‌ای از الگوریتم‌های مرتب‌سازی (Counting sort) مرتب‌سازی شمارشی



Counting sort	Quick sort	Merge sort	Insertion sort & Bubble sort & Selection sort	شاخص
غیرمقایسه‌ای	مقایسه‌ای	مقایسه‌ای	مقایسه‌ای	Comparative (مقایسه‌ای)
برون‌جا	درجا/برون‌جا	برون‌جا	درجا	In-place (درجا)
داخلی	داخلی	داخلی	داخلی	Internal (داخلی)
ناپایدار	ناپایدار	پایدار	پایدار/ناپایدار	Stable (پایدار)
$O(n + k)$	$O(n^2)$ $O(n \log n)$ $O(n \log n)$	$O(n \log n)$	$O(n^2)$	Worst-case order

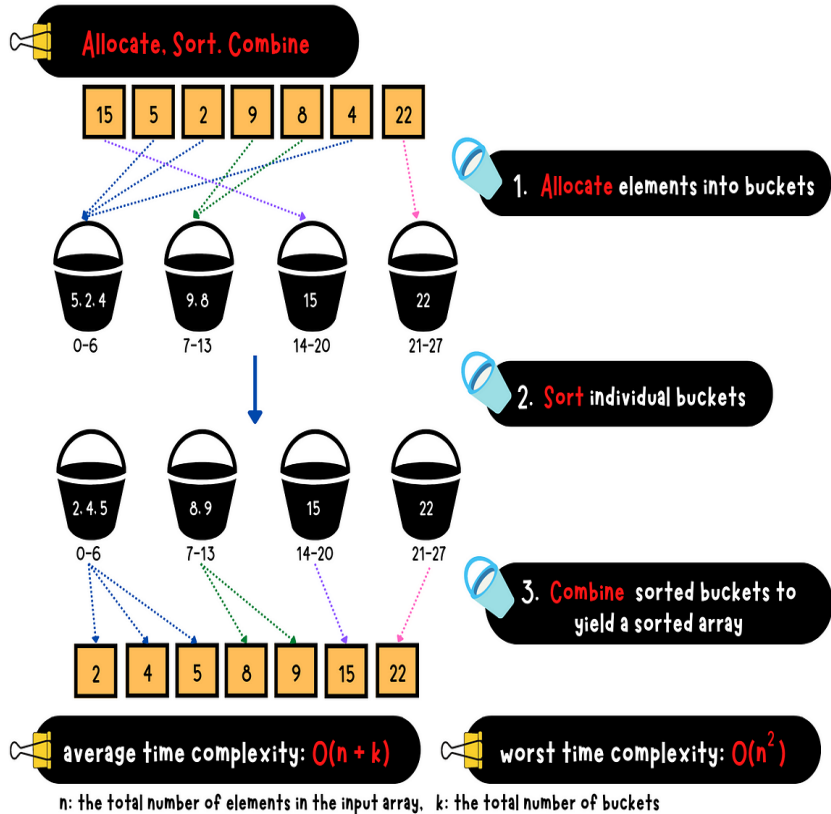
خلاصه‌ای از الگوریتم‌های مرتب‌سازی مرتب‌سازی پایه‌ای (Radix sort)



Radix sort	Counting sort	Quick sort	Merge sort	Insertion sort & Bubble sort & Selection sort	شاخص
غیر مقایسه‌ای	غیر مقایسه‌ای	مقایسه‌ای	مقایسه‌ای	مقایسه‌ای	Comparative (مقایسه‌ای)
برون‌جا	برون‌جا	درجا/برون‌جا	برون‌جا	درجا	In-place (درجا)
داخلی	داخلی	داخلی	داخلی	داخلی	Internal (داخلی)
پایدار	ناپایدار	ناپایدار	پایدار	پایدار/ناپایدار	Stable (پایدار)
$O(k(n + r))$	$O(n + k)$	$O(n^2)$ $O(n \log n)$ $O(n \log n)$	$O(n \log n)$	$O(n^2)$	Worst-case order

خلاصه‌ای از الگوریتم‌های مرتب‌سازی مرتب‌سازی سطلی (Bucket sort)

Bucket Sort Algorithm

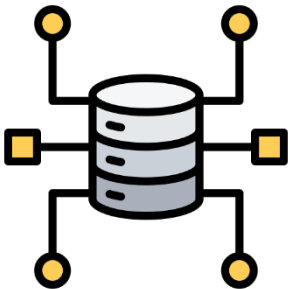


Bucket sort	Radix sort	Counting sort	Quick sort	Merge sort	Insertion & Bubble & Selection sort	شاخص
غیر مقایسه‌ای	غیر مقایسه‌ای	غیر مقایسه‌ای	مقایسه‌ای	مقایسه‌ای	مقایسه‌ای	Comparative (مقایسه‌ای)
برون‌جا	برون‌جا	برون‌جا	درجا/برون‌جا	برون‌جا	درجا	In-place (درجا)
داخلی	داخلی	داخلی	داخلی	داخلی	داخلی	Internal (داخلی)
پایدار	پایدار	ناپایدار	ناپایدار	پایدار	پایدار ناپایدار	Stable (پایدار)
$O(n)$	$O(k(n + r))$	$O(n + k)$	$O(n^2)$ $O(n \log n)$ $O(n \log n)$	$O(n \log n)$	$O(n^2)$	Worst-case order

سوال سوم

برای هر یک از شرایط زیر، یک الگوریتم sort انتخاب کنید. (selection sort, insertion sort, merge sort)

- فرض کنید یک ساختار داده داریم که یک منظومه ای از داده ها را در خود نگه داری می کند، این ساختار داده دو دستور `get_at(i)` و `set_at(i,x)` را به ترتیب از $O(1)$ و $O(n \log n)$ پشتیبانی می کند. بهترین الگوریتمی را انتخاب کنید که به صورت in-place داده ها را در این مجموعه داده ذخیره کند.

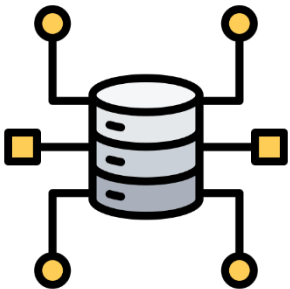


سوال سوم

برای هر یک از شرایط زیر، یک الگوریتم sort انتخاب کنید. (selection sort, insertion sort, merge sort)

- فرض کنید یک ساختار داده داریم که یک منظومه ای از داده ها را در خود نگه داری می کند، این ساختار داده دو دستور `get_at(i)` و `set_at(i,x)` را به ترتیب از $O(1)$ و $O(n \log n)$ پشتیبانی می کند. بهترین الگوریتمی را انتخاب کنید که به صورت in-place داده ها را در این مجموعه داده ذخیره کند.

Answer: Selection sort



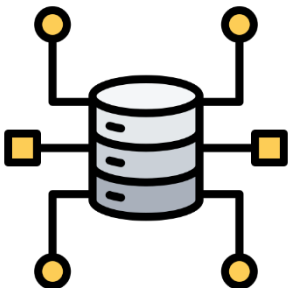
سوال سوم

برای هر یک از شرایط زیر، یک الگوریتم sort انتخاب کنید. (selection sort, insertion sort, merge sort)

- فرض کنید یک ساختار داده داریم که یک منظومه ای از داده ها را در خود نگه داری می کند، این ساختار داده دو دستور `get_at(i)` و `set_at(i,x)` را به ترتیب از $O(1)$ و $O(n \log n)$ پشتیبانی می کند. بهترین الگوریتمی را انتخاب کنید که به صورت in-place داده ها را در این مجموعه داده ذخیره کند.

Answer: Selection sort

- فرض کنید یک آرایه از پوینترها داریم که به تعدادی object قابل مقایسه اشاره می کنند، مقایسه هر دو object از $O(\log n)$ انجام پذیر است. الگوریتمی را انتخاب کنید که این آرایه را به طور non-decreasing در سریعترین زمان مرتب کند.



سوال سوم

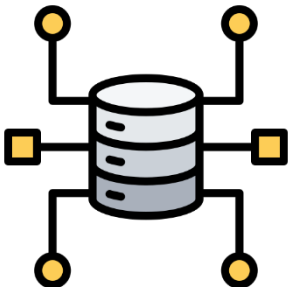
برای هر یک از شرایط زیر، یک الگوریتم sort انتخاب کنید. (selection sort, insertion sort, merge sort)

- فرض کنید یک ساختار داده داریم که یک منظومه ای از داده ها را در خود نگه داری می کند، این ساختار داده دو دستور `get_at(i)` و `set_at(i,x)` را به ترتیب از $O(1)$ و $O(n \log n)$ پشتیبانی می کند. بهترین الگوریتمی را انتخاب کنید که به صورت in-place داده ها را در این مجموعه داده ذخیره کند.

Answer: Selection sort

- فرض کنید یک آرایه از پوینترها داریم که به تعدادی object قابل مقایسه اشاره می کنند، مقایسه هر دو object از $O(\log n)$ انجام پذیر است. الگوریتمی را انتخاب کنید که این آرایه را به طور non-decreasing در سریعترین زمان مرتب کند.

Answer: Merge sort



سوال سوم

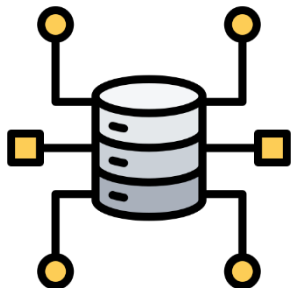
برای هر یک از شرایط زیر، یک الگوریتم sort انتخاب کنید. (selection sort, insertion sort, merge sort)

- فرض کنید یک ساختار داده داریم که یک منظومه ای از داده ها را در خود نگه داری می کند، این ساختار داده دو دستور `get_at(i)` و `set_at(i,x)` را به ترتیب از $O(1)$ و $O(n \log n)$ پشتیبانی می کند. بهترین الگوریتمی را انتخاب کنید که به صورت in-place داده ها را در این مجموعه داده ذخیره کند.

Answer: Selection sort

- فرض کنید یک آرایه از پوینترها داریم که به تعدادی object قابل مقایسه اشاره می کنند، مقایسه هر دو object از $O(\log n)$ انجام پذیر است. الگوریتمی را انتخاب کنید که این آرایه را به طور non-decreasing در سریعترین زمان مرتب کند.

Answer: Merge sort



- فرض کنید یک آرایه مرتب شده از n عدد integer داریم، یک الگوریتم مشخص، بعضی از اعضای مجاور در این آرایه را مجموعاً $\log \log n$ بار جابجا کرده است. الگوریتمی را انتخاب کنید که این آرایه خاص را در سریعترین حالت ممکن مرتب کند.

سوال سوم

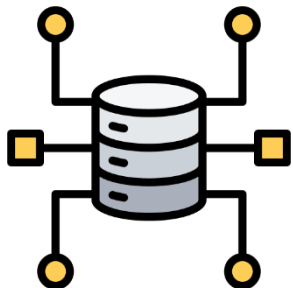
برای هر یک از شرایط زیر، یک الگوریتم sort انتخاب کنید. (selection sort, insertion sort, merge sort)

- فرض کنید یک ساختار داده داریم که یک منظومه ای از داده ها را در خود نگه داری می کند، این ساختار داده دو دستور $get_at(i)$ و $set_at(i,x)$ را به ترتیب از $O(1)$ و $O(n \log n)$ پشتیبانی می کند. بهترین الگوریتمی را انتخاب کنید که به صورت in-place داده ها را در این مجموعه داده ذخیره کند.

Answer: Selection sort

- فرض کنید یک آرایه از پوینترها داریم که به تعدادی object قابل مقایسه اشاره می کنند، مقایسه هر دو object از $O(\log n)$ انجام پذیر است. الگوریتمی را انتخاب کنید که این آرایه را به طور non-decreasing در سریعترین زمان مرتب کند.

Answer: Merge sort



- فرض کنید یک آرایه مرتب شده از n عدد integer داریم، یک الگوریتم مشخص، بعضی از اعضای مجاور در این آرایه را مجموعاً $\log \log n$ بار جابجا کرده است. الگوریتمی را انتخاب کنید که این آرایه خاص را در سریعترین حالت ممکن مرتب کند.

Answer: Insertion sort

سوال چهارم

تعداد n چاه نفت در یک نقشه دو بعدی داریم. چاه نفت i ام در مختصات x_i و y_i قرار دارد. می خواهیم یک لوله افقی اصلی با مختصات $y = c$ از بین این چاه ها بگذرانیم و هر چاه را با یک لوله عمودی به این لوله افقی متصل کنیم. در زمان میانگین $O(n)$ مقدار c را طوری تعیین کنید که مجموع طول لوله های عمودی کمینه باشد.



سوال چهارم

تعداد n چاه نفت در یک نقشه دو بعدی داریم. چاه نفت i ام در مختصات x_i و y_i قرار دارد. می‌خواهیم یک لوله افقی اصلی با مختصات $y = c$ از بین این چاه‌ها بگذرانیم و هر چاه را با یک لوله عمودی به این لوله افقی متصل کنیم. در زمان میانگین $O(n)$ مقدار c را طوری تعیین کنید که مجموع طول لوله‌های عمودی کمینه باشد.

اگر n فرد باشد، مختصات y خط لوله اصلی را برابر میانه مختصات تمام چاه‌ها انتخاب می‌کنیم به. اگر n زوج باشد، ما می‌توانیم مختصات y خط لوله را هر چیزی بین $\lfloor \frac{n+1}{2} \rfloor$ و $\lceil \frac{n+1}{2} \rceil$ انتخاب کنید.



سوال پنجم

می‌خواهیم از بین n عدد، k امین کوچکترین عنصر، $2k$ امین کوچکترین عنصر و به همین ترتیب تا $\lfloor \frac{n}{k} \rfloor$ امین کوچکترین عنصر را پیدا کنیم. روشی از $O(n \log \frac{n}{k})$ برای این کار ارائه دهید.



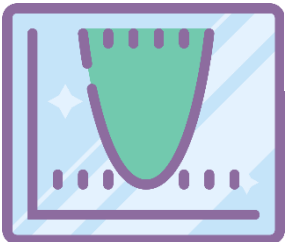
سوال پنجم

می‌خواهیم از بین n عدد، k امین کوچکترین عنصر، $2k$ امین کوچکترین عنصر و به همین ترتیب تا $\lfloor \frac{n}{k} \rfloor$ امین کوچکترین عنصر را پیدا کنیم. روشی از $O(n \log \frac{n}{k})$ برای این کار ارائه دهید.

پاسخ:

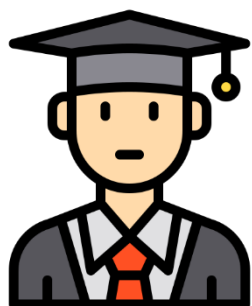
برای حالتی که $k < n/k$ باشد به این صورت عمل می‌کنیم:

ابتدا ارایه را به k بخش با n/k عضو تقسیم می‌کنیم و هر بخش را با زمان اجرای $O(n/k * \log(n/k))$ مرتب می‌کنیم. مرتب سازی همه بخش‌ها زمان اجرای $O(n * \log(n/k))$ خواهد داشت. حال همه این k بخش را در نظر می‌گیریم، عنصر مینیمم کل در ابتدای یکی از این k بخش است (چون این بخش‌ها مرتب شده هستند). یک min heap در نظر می‌گیریم و k عنصر ابتدای این k بخش را درون این min heap با زمان اجرای $O(k)$ می‌گذاریم (عملیات min heapify) و آن k عنصر را از بخش‌های n/k عنصری حذف می‌کنیم. سپس در k مرحله عنصر مینیمم درون heap را حذف کرده و به جای آن مینیمم عنصری که الان در ابتدای k بخش هستند را به heap اضافه می‌کنیم. $O(k \log k)$ پس از k مرحله عنصری که مینیمم در heap است k امین کوچکترین عنصر است. همین k مرحله را n/k بار تکرار می‌کنیم تا همه عناصر خواسته شده به ترتیب به دست بیاید. $O(n \log(k))$. طبق فرض که $k < n/k$ بود زمان اجرای این الگوریتم $O(n \log(k))$ خواهد بود.



سوال ششم

دانش‌آموزان یک کلاس را می‌خواهیم با توجه به قدشان به دو گروه تقسیم کنیم به طوری که اعضای گروه اول از همه‌ی اعضای گروه دوم کوتاه‌قدتر باشند. می‌خواهیم گروه اول تا جای ممکن کوچک باشد ولی از طرفی می‌خواهیم مجموع قد افراد گروه اول حداقل برابر نصف مجموع قد همه‌ی افراد باشد. روشی ارائه دهید که در $O(n)$ این کار را انجام دهد.



سوال ششم

دانش‌آموزان یک کلاس را می‌خواهیم با توجه به قدشان به دو گروه تقسیم کنیم به طوری که اعضای گروه اول از همه‌ی اعضای گروه دوم کوتاه‌تر باشند. می‌خواهیم گروه اول تا جای ممکن کوچک باشد ولی از طرفی می‌خواهیم مجموع قد افراد گروه اول حداقل برابر نصف مجموع قد همه‌ی افراد باشد. روشی ارائه دهید که در $O(n)$ این کار را انجام دهد.

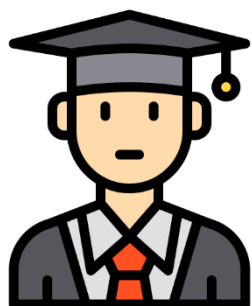
پاسخ:

ابتدا مجموع قد تمام افراد کلاس را با زمان اجرای $O(n)$ پیدا می‌کنیم (مثلاً S). سپس میانه قد افراد کلاس را با $O(n)$ پیدا می‌کنیم. حال مجموع قد تمام افرادی که قد کمتر از میانه دارند را پیدا می‌کنیم. اگر این مقدار بیشتر از $S/2$ بود یعنی باید افرادی به این افراد اضافه شوند و اگر کمتر بود یعنی باید افرادی از این افراد کم شوند. به همین ترتیب به صورت بازگشتی میانه افراد بلندتر از میانه (یا کوتاه‌تر از میانه) را با $O(n/2)$ پیدا می‌کنیم و این کار را تا جایی تکرار می‌کنیم که به مجموعه‌ای از افراد برسیم که متناسب با خواسته سوال است.

زمان اجرا: $O(n + n + n/2 + n/2 + n/4 + n/4 + \dots)$

می‌دانیم: $n + n/2 + n/4 + \dots < 2n$

پس زمان اجرای این الگوریتم معادل $O(4n)$ یا همان $O(n)$ است.



سوال هفتم

جدول درهم‌سازی با اندازه‌ی ۱۰ (خانه‌های شماره‌ی صفر تا ۹) زیر را در نظر بگیرید که از آدرس‌دهی باز براساس تابع درهم‌سازی $h(i) = i \bmod 10$ و Linear Probing استفاده می‌کند. بعد از درج ۶ کلید، محتوای این جدول به صورت زیر خواهد بود:

۰	۱	۲	۳	۴	۵	۶	۷	۸	۹
		۴۲	۲۳	۳۴	۵۲	۴۶	۳۳		

چه تعداد از دنباله‌های متفاوت درج کلیدها در این جدول وجود دارد که جدول بالا را نتیجه می‌دهد؟



سوال هفتم

جدول درهم‌سازی با اندازه‌ی ۱۰ (خانه‌های شماره‌ی صفر تا ۹) زیر را در نظر بگیرید که از آدرس‌دهی باز براساس تابع درهم‌سازی $h(i) = i \bmod 10$ و Linear Probing استفاده می‌کند. بعد از درج ۶ کلید، محتوای این جدول به صورت زیر خواهد بود:

۰	۱	۲	۳	۴	۵	۶	۷	۸	۹
		۴۲	۲۳	۳۴	۵۲	۴۶	۳۳		

چه تعداد از دنباله‌های متفاوت درج کلیدها در این جدول وجود دارد که جدول بالا را نتیجه می‌دهد؟

در این جدول، کلیدهای ۴۲، ۲۳، ۳۴ و ۴۶ در مکان صحیح خود هستند و با هر ترتیبی می‌توانند درج شده باشند. ولی کلید ۵۲ باید حتماً بعد از عناصر ۴۲، ۲۳ و ۳۴ درج شده باشد، چون در زمان درج آن این مکان‌ها پر بوده است که ۵۲ در خانه‌ی ۵ قرار گرفته است. همچنین کلید ۳۳ نیز باید بعد از تمامی کلیدها درج شده باشد. بنابراین کل تعداد دنباله‌های مختلف درج برابر $3! \times 5 = 30$ می‌باشد.



سوال هشتم

- الگوریتمی از $O(n)$ طراحی کنید، که یک آرایه با عناصر مثبت و یک عدد S را ورودی بگیرد، مشخص کند که آیا دو عدد در این آرایه وجود دارد که مجموع آن‌ها S شود یا خیر؟



سوال هشتم

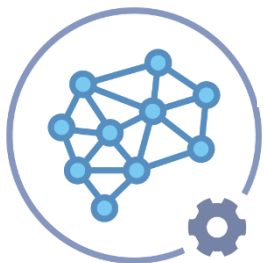
- الگوریتمی از $O(n)$ طراحی کنید، که یک آرایه با عناصر مثبت و یک عدد S را ورودی بگیرد، مشخص کند که آیا دو عدد در این آرایه وجود دارد که مجموع آن‌ها S شود یا خیر؟

روی عناصر آرایه پیمایش می‌کنیم و به ازای هر عنصری که می‌بینیم، آن را در یک جدول درهم سازی ذخیره

می‌کنیم و سپس وجود مقدار $S - arr[i]$ را در جدول درهم سازی جستجو می‌کنیم. این روند را تا جایی

ادامه می‌دهیم که مقدار $S - arr[i]$ در جدول موجود باشد. هزینه اضافه کردن و جستجو در جدول درهم

سازی $O(1)$ است و هزینه پیمایش روی آرایه $O(n)$ است، بنابراین هزینه کلی این الگوریتم $O(n)$ است.



راهنمایی درباره تمرین:

۱.

نمره

الگوریتمی طراحی کنید که با گرفتن یک آرایه که در آن هر عنصر حداکثر ۲۰ عنصر سمت چپ خود دارد که از او بزرگ‌تر باشند، این آرایه را در زمان $O(n)$ مرتب کند.

۲.

نمره

یک الگوریتم با مرتبه زمانی $O(n \lg k)$ برای ادغام k لیست مرتب شده در یک لیست مرتب شده، که در آن n تعداد کل عناصر در همه لیست‌های ورودی است، ارائه دهید.

۳.

نمره

محمد فکر می‌کند که اگر روش زنجیره‌سازی مجزا (*separate chaining*) برای ساخت یک *hash table* را اینگونه تغییر دهد که در روش جدید هر لیست پیوندی در یک خانه آرایه به صورت مرتب شده باشد، نتیجتاً به کارایی (*performance*) بهتری خواهد رسید. در روش جدید محمد پیچیدگی زمانی برای جستجوی موفق، جستجوی ناموفق، درج و حذف را محاسبه کنید.



راهنمایی درباره تمرین:

۴.

نمره

الگوریتم *Quick Sort* شامل دو فراخوان بازگشتی به خود است. پس از آنکه *Quick Sort*، *Partition* را فراخوانی می کند، زیرآرایه سمت چپ را به صورت بازگشتی مرتب می کند و سپس زیرآرایه سمت راست را به صورت بازگشتی مرتب می کند. فراخوانی بازگشتی دوم در *Quick Sort* واقعاً ضروری نیست و می توان با استفاده از یک ساختار کنترل تکرار شونده از آن اجتناب کرد. این تکنیک *tail recursion* نامیده می شود. نسخه زیر از *Quick Sort* را در نظر بگیرید که *tail recursion* را شبیه سازی می کند:

TAIL-RECURSIVE-QUICKSORT(A, p, r)

```
1  while  $p < r$ 
2      // Partition and sort left subarray.
3       $q = \text{PARTITION}(A, p, r)$ 
4      TAIL-RECURSIVE-QUICKSORT( $A, p, q - 1$ )
5       $p = q + 1$ 
```

ثابت کنید که $\text{TAIL-RECURSIVE-QUICKSORT}(A, 1, A.length)$ به درستی آرایه A را مرتب می کند.



راهنمایی درباره تمرین:

۵.

نمره

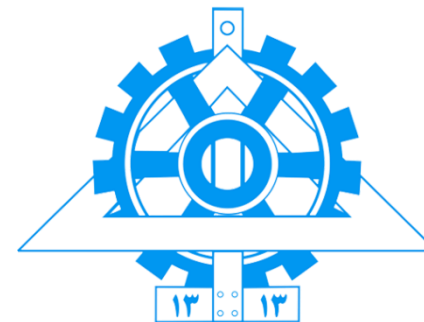
الگوریتمی طراحی کنید که با گرفتن n عدد صحیح بین 0 تا k پیش‌پردازشی روی ورودی انجام داده و سپس در زمان $O(1)$ با گرفتن دو عدد a و b مشخص کند که چه تعداد از اعداد ورودی در بازه $[a, \dots, b]$ هستند. پیش‌پردازشی که الگوریتم روی ورودی انجام می‌دهد باید با پیچیدگی زمانی $\Theta(n + k)$ باشد.

۶.

نمره

الگوریتمی با زمان اجرای $O(n)$ طراحی کنید که با گرفتن مجموعه S شامل n عدد یکتا و عدد مثبت k ، $k \leq n$ تا عددی را در S مشخص کند که نزدیک‌ترین به میانه S هستند.





لطفا در صورت هر گونه ابهام یا پرسش، سوالات خود را از طریق ایمیل با دستیاران آموزشی این مبحث، در ارتباط بگذارید.

ساختمان داده و الگوریتم - پائیز ۱۴۰۲

Email: MohammadAmanlou2@gmail.com
phateme.k@gmail.com

